gzip and the DEFLATE Compression Algorithm

Gian Fernadez, Henry Hodge,
Jackson Weil
4/21/2026
COSC 581

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Questions

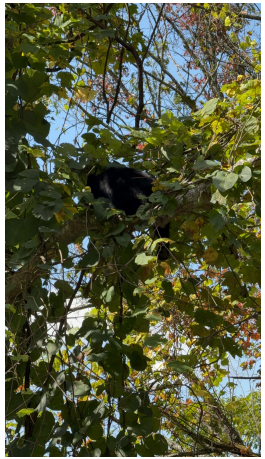
1. Where does the name LZ77 come from?
2. What does the `g` in `gzip` stand for?
3. Who created DEFLATE?

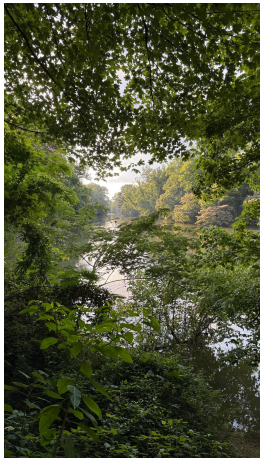
https://forms.gle/ha7yWhc7YJC4jQai7

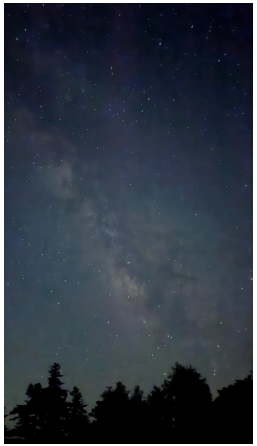
Presenters

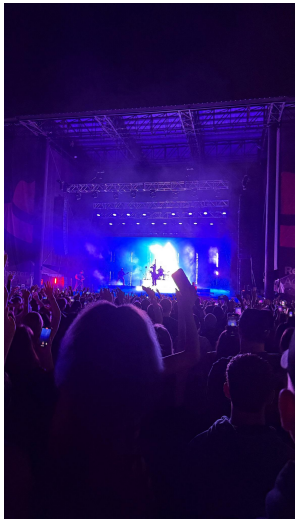
Gian Fernandez

- Undergraduate CS major
- Born in California, moved to Tennessee in 2005
- Like to hike



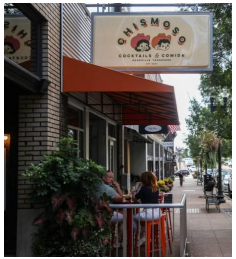






Henry Hodge

- Masters CS Student, TENNLab, Dr. Schuman
- Goals? Not sure, but I like theoretical stuff
- Born in Nashville, but from Knoxville

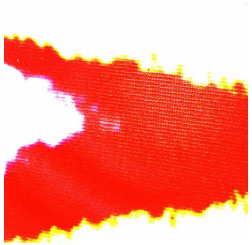


KOKO
SIPS + BITES

SmashCity Knoxville seized for unpaid taxes as owners await trial on theft charges

Henry Hodge

- My cat's name is Crash
- I enjoy soccer, tennis, and boxing
- Music
- I tore my ACL



Jackson Weil

- Master CS Student
- TA for 302
- From Nashville
- In a chess phase
- Enjoy tennis and basketball
- Rocket League
- One Piece

Outline

1. Overview of gzip and DEFLATE
2. History
    a. Algorithmic Development
    b. Popular tooling: gzip
3. Algorithms
    a. DEFLATE
    b. LZ77
    c. Huffman Encoding
4. Applications
5. Open Issues
6. References
7. Discussion
8. Questions, again

Overview of `gzip`

Developed as part of the GNU project, bypassing patents

A file format…

- 10 byte header
- DEFLATE compressed data
- 8 byte footer

And a command line tool…

- Compresses and converts to format
- Commonly used with tar

```
% gzip junk.txt
% tar -czf archive.tar.gz /path/to/directory
```

Overview of `gzip`

Command invocation, not code.

```
% gzip junk.txt
% tar -czf archive.tar.gz /path/to/directory
```

Overview of DEFLATE

Lossless compression algorithm

Combines LZ77 and Huffman coding

Used in zlib format, gzip format, PNG file, ZIP file format

Takes any sequence of bytes, and outputs sequence of blocks, which may be compressed

Created by Phil Katz

History

History of `gzip`

Developed in the 1990's for the GNU Project as a patent-free alternative meant to replace Unix's "compress" tool (IBM had the patents for the LZW algorithm)

The g in gzip stands for GNU

Jean-loup Gailly, left - compression

Mark Adler, right - decompression

History of DEFLATE

Developed by Phil Katz in 1990 as an efficient, patent-free compression algorithm

It combined two older ideas: LZ77-style compression and Huffman coding

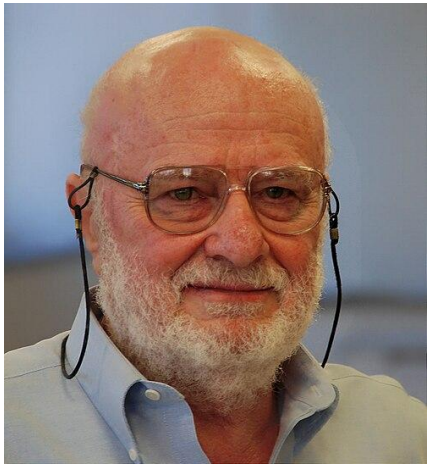
DEFLATE became widely adopted and became the core compression method in gzip, zlib and ZIP

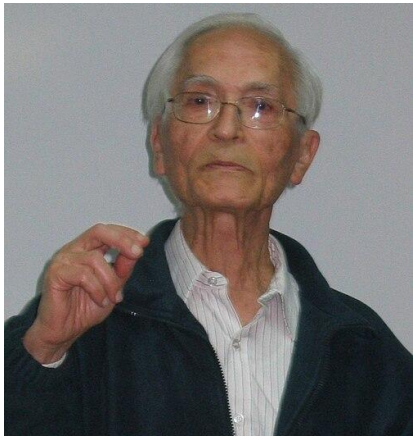
History of LZ77

Introduced in 1977 by Abraham Lempel and Jacob Ziv in their paper, "*A Universal Algorithm for Sequential Data Compression*"

Introduced the idea of a sliding window dictionary

Replaced repeated data with length-distance references



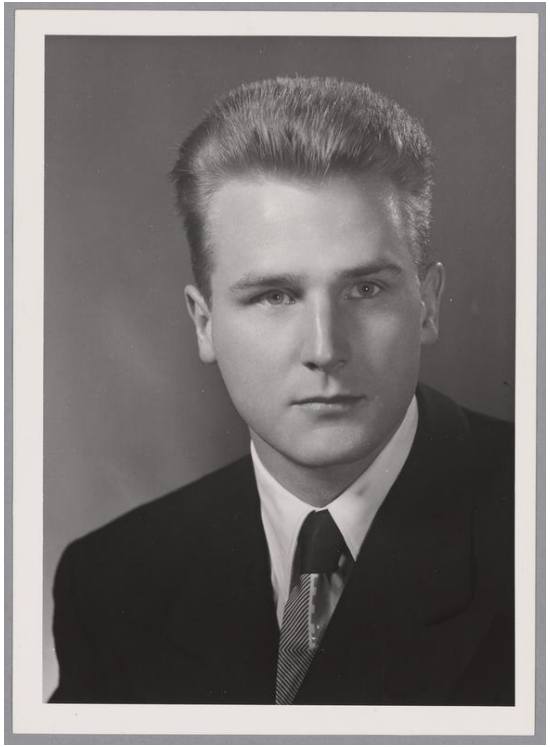


History of Huffman Coding

Developed in 1952 by David A. Huffman while he was a student at MIT

It produces optimal prefix codes by assigning shorter codes to more frequent symbols

This reduces the total number of bits needed by taking advantage of symbol frequencies



Algorithms

Algorithms: LZ

Lempel-Ziv 1977: this is what the name LZ77 is for

Lempel-Ziv-Storer-Szymanski (LZSS) is a variant used here

Lossless compression algorithm

Replaces repeated sequences with back pointers, or *length-distance* pairs

Uses a sliding window as a dictionary, selects longest matching substring

Implementations vary

- Speed vs. quality tradeoff
  - Hash chaining
  - BST

Algorithms: LZ Example

Original Data

```
A ␣ S A L A D ; ␣ A ␣ S A L S A ;
```

Encoded Data

```
A ␣ S A L A D ; ␣ 5:9 2:3 ;
```

[2]

- Length 5, Distance 9: **A SAL**
- Length 2, Distance 3: **SA**

Algorithms: Huffman

Achieves optimal prefix codes for any input

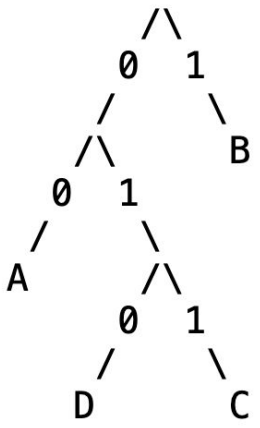
DEFLATE supports fixed and dynamic Huffman coding

- Fixed uses a fixed Huffman tree
- Dynamic builds fresh trees per block (discussed shortly)
- Code lengths are max. 15 bits

How to efficiently transmit tree?

- Restrictions:
  - Shorter codes lexicographically precede longer codes
  - Codes of same length follow lexicographic ordering of symbols

Algorithms: Huffman Example



[1]

Symbol	Code
A	00
B	1
C	011
D	010

Apply Restrictions

*unique

Symbol	Code
A	10
B	0
C	110
D	111

[1]

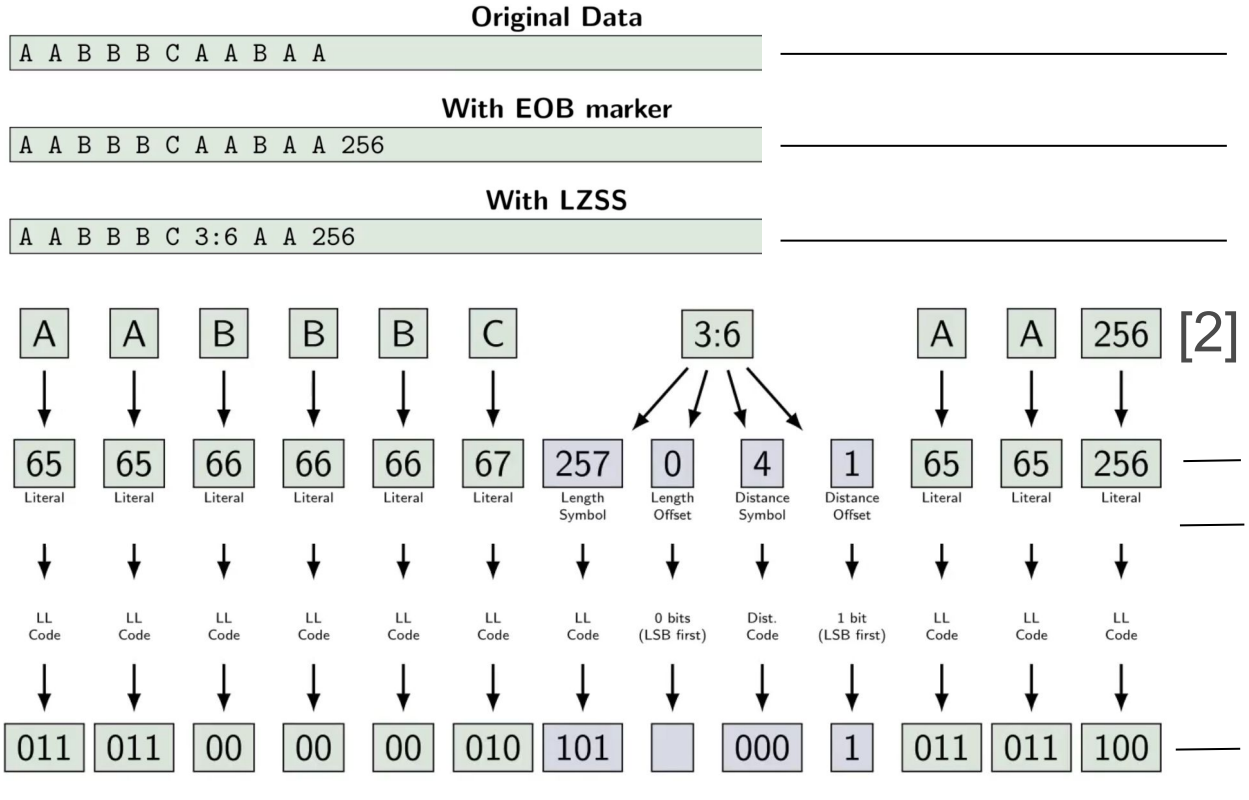
Bit length representation: (A,B,C,D) -> **(2,1,3,3)**

Algorithms: DEFLATE

Outputs a sequence of compressed blocks:

1. Optionally run LZ on input
2. Partition into *blocks* ad hoc
3. Run Huffman on the block, depending on type

Algorithms: DEFLATE Example (type 1,2)



1. Original Data (block)
2. Append EOB marker
3. Apply LZSS
4. Convert data to literal-length alphabet
5. Generate Huffman code for block (or used fixed tree)
   - Use one tree for literal-length alphabet (0-285)
   - One tree for distance coding
6. Encode data
7. Prepend block metadata

Algorithms: DEFLATE

Outputs a sequence of compressed blocks:

1. Optionally run LZ on input
2. Partition into *blocks* ad hoc
3. Run Huffman on the block, depending on type

Each block has a type, and a bit to signify whether it is the final block.

Blocks end with EOB marker (End Of Block = 256)

Block types:

- Type 0: no compression
- Type 1: fixed Huffman
  - Fixed tree, excluded from compressed representation
- Type 2: dynamic Huffman
  - 2 new trees for each block - max code length of 15
    - Literal length tree
    - Distance tree
  - Bit length representations of trees are then huffman coded again
  - So each block has three Huffman trees

Algorithms: DEFLATE Block 2 alphabets

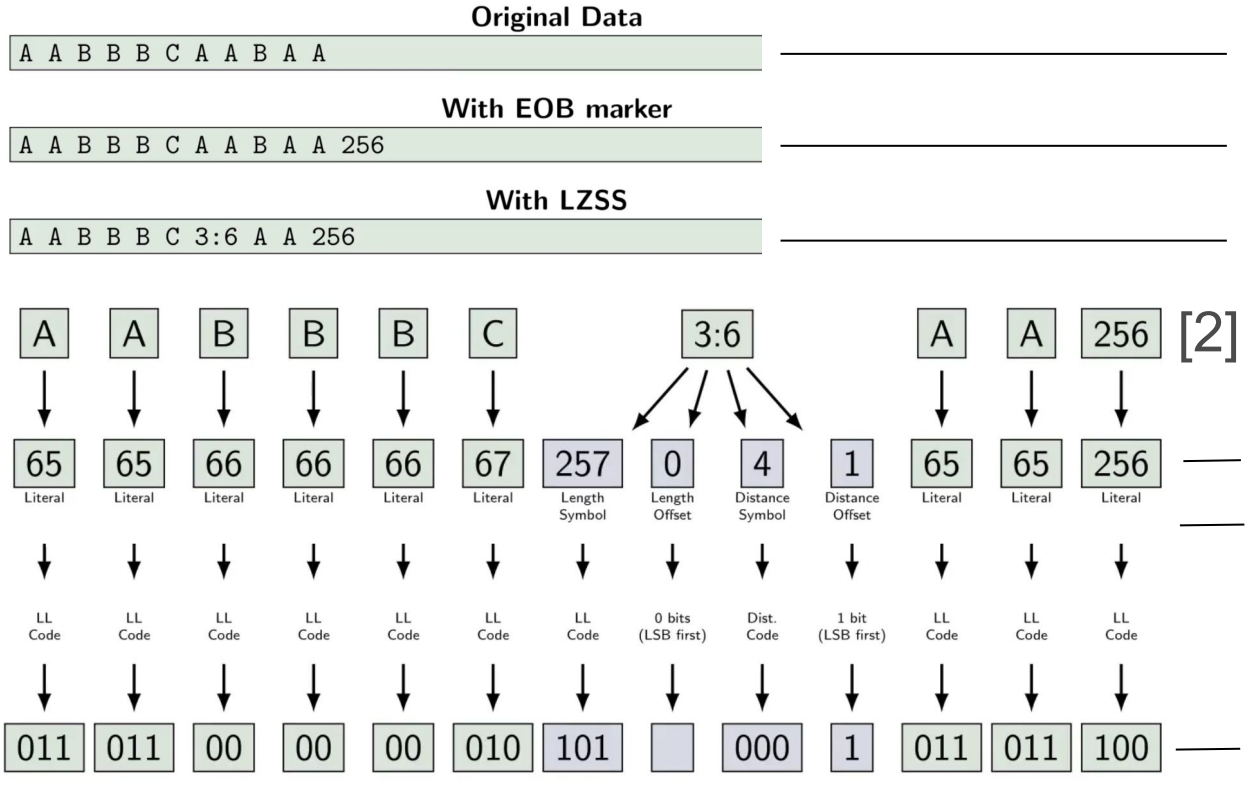
[0..285]: literal-length alphabet

- [0..255]: literals
- {256}: EOB marker
- [257..285]: possible length symbols/ranges
  - Many symbols correspond to a single length in *length-distance* pair
  - Some correspond to ranges, and include offset bits

[1..]: distance alphabet

- Contains all the unique distances
- Some symbols correspond to single distances, others correspond to ranges

Algorithms: DEFLATE Example (type 1,2)



1. Original Data (block)
2. Append EOB marker
3. Apply LZSS
4. Convert data to literal-length alphabet
5. Generate Huffman code for block (or used fixed tree)
   - Use one tree for literal-length alphabet (0-285)
   - One tree for distance coding
6. Encode data
7. Prepend block metadata

Algorithms: DEFLATE Example

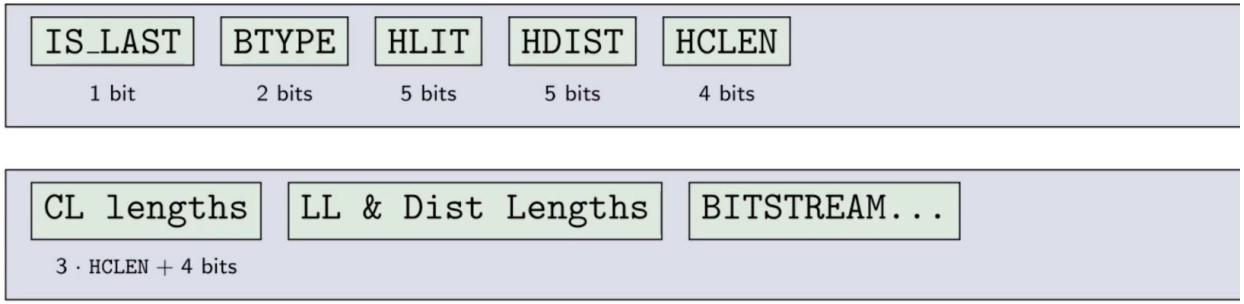
Type 0, 1

- Do not require tree transmission

[2]

Algorithms: DEFLATE Example

Type 2



HLIT: # of code lengths > 257

HDIST: # of code lengths for distances > 1

HCLEN: # of code lengths for LL and Dist lengths > 4

Code length tree
- encodes the LL and Dist trees

Literal-Length and Distance trees
- Encoded by CL
- List of code lengths

[2]

Offers a compact representation of prefix coding trees to reduce overhead

Applications

Application 1: Digital Archiving

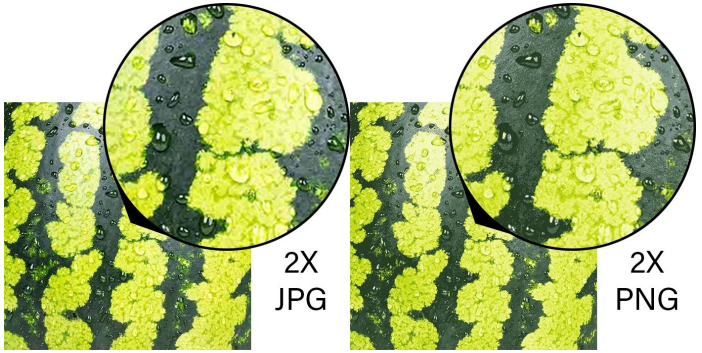
Used to store large collections of digital files (documents, records, etc)

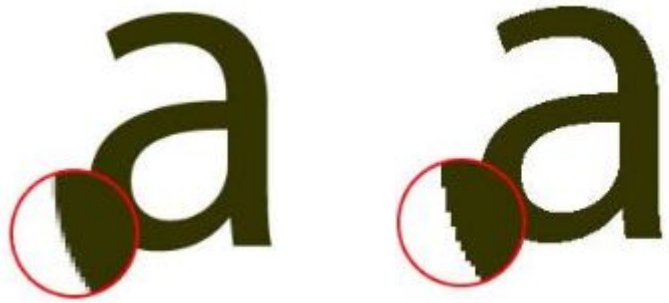
Reduces the need for physical storage space while preserving exact original data

Super important for backups and historical data where loss or corruption of data would be unacceptable

Application 2: PNG

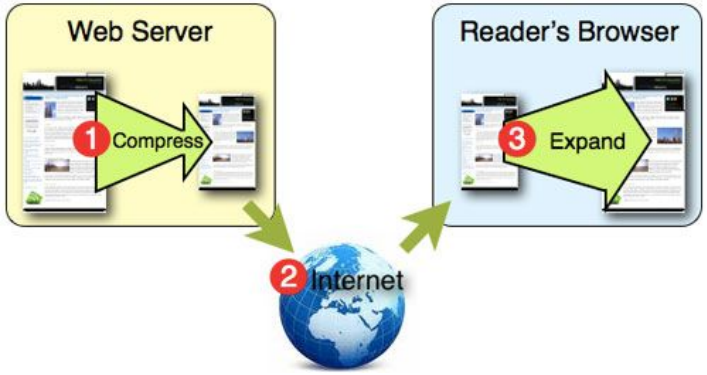
- Image files are large
- PNG uses DEFLATE
- Every pixel is preserved
- JPG uses lossy compression





Application 3: Web Compression

- DEFLATE (via gzip) can compress HTML 70-90%
- Servers send compressed versions
- Browsers are designed to decompress on arrival
- This results in less data traveling across the web



Open Issues

Limited Window Size

- LZ77 uses a fixed sliding window which makes long-distance repetitions hard to detect

Suboptimal compared to theoretical limits

- DEFLATE is efficient but it doesn’t reach the entropy limit

LZ77 picks local matches, not globally optimal ones

References

[1] L. P. Deutsch, “rfc1951,” *datatracker.ietf.org*, 1996. https://datatracker.ietf.org/doc/html/rfc1951

[2] B. Bird, “- YouTube,” *www.youtube.com*, May 23, 2023. https://www.youtube.com/watch?v=SJPvNi4HrWQ&t=7409s (accessed Apr. 21, 2026).

[3] J. Ziv and A. Lempel, “A Universal Algorithm for Sequential Data Compression,” IEEE Transactions on Information Theory, 1977.

[4] J.-L. Gailly and M. Adler, “gzip file format specification,” *gnu.org*, 1992. https://www.gnu.org/software/gzip/manual/gzip.html

[5] “Deflate - Wikipedia,” en.wikipedia.org. https://en.wikipedia.org/wiki/Deflate (accessed Mar. 10, 2026).

[6] R. Holmes, “Google’s Need for Speed – How To Compress Your Site,” Pilot Digital, Apr. 26, 2010. https://pilotdigital.com/blog/googles-need-for-speed-how-to-compress-your-site/ (accessed Apr. 3, 2026).

Discussion

Questions

1. Where does the name LZ77 come from?
2. What does the `g` in `gzip` stand for?
3. Who created DEFLATE?

https://forms.gle/ha7yWhc7YJC4jQai7